UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/976,726	10/12/2001	Jason King	5150-58700	2464

35690 7590 12/06/2004

MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL, P.C.
P.O. BOX 398
AUSTIN, TX 78767-0398

EXAMINER	
ZHOU, TING	

ART UNIT	PAPER NUMBER
2173	

DATE MAILED: 12/06/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/976,726	KING ET AL.	
	Examiner	Art Unit	
	Ting Zhou	2173	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *02 September 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-66* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-66* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *9/2/04*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. The amendment filed on 2 September 2004 have been received and entered. Claims 1-66 as amended are pending in the application.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

2. Claims 1-7 and 10-66 are rejected under 35 U.S.C. 102(b) as being anticipated by Kodosky et al. U.S. Patent 5,301,336.

Referring to claims 1, 19, 23 and 32, Kodosky et al. teach a method and memory medium comprising creating a graphical user interface for the graphical program in response to user input (creation of a graphical program representing a virtual instrument on a graphical user interface that permits interaction with the user)(column 17, lines 48-61 and Figure 22); displaying an event structure node in a block diagram for the graphical program in response to user input (user designing and interacting with the block diagram, which is made up of connected nodes, or icons, as shown in Figure 22) (column 15, lines 54-63, column 34, lines 27-41 and column 38, lines 5-18); and configuring the event structure node to receive and respond to one or more user interface events during execution of the graphical program (the user can interact with the block diagram by utilizing icons to build the block diagram; furthermore, as an example, the user

interface event "CLEAR", shown in Figure 25, can be used to remove certain wires from the block diagram) (column 9, lines 56-64, column 17, line 48 – column 18, line 32 and column 19, lines 11-15). This is further recited in column 8, lines 37-57, column 14, lines 10-27 and column 50, lines 4-40.

Referring to claims 36, 53 and 66, Kodosky et al. teach a method and memory medium comprising a memory storing program instructions, a processor coupled to the memory and a display device, wherein the processor is operable to execute the program instructions stored in memory (column 10, lines 10-18 and column 50, lines 4-7) to display a first node in a block diagram of the graphical program in response to user input (user selecting an icon to use in building the block diagram) (column 9, lines 56-64) and associate graphical source code (a plurality of interconnected nodes and icons) (Figures 16 and 22) with the first node in response to user input (a node or icon making up a block diagram can contain sub-diagrams containing a plurality of interconnected nodes and icons) (column 8, lines 45-57), wherein the graphical source code is operable to execute in response to a user interface event (each virtual instrument graphical program, which can contain a plurality of interconnected nodes, are configured to produce an output when a certain input is received, for example, when a certain user input is received) (column 8, lines 45-57 and column 26, lines 54-64). This is further recited in column 14, lines 10-27.

Referring to claims 2, 20 and 33, Kodosky et al. teach the event structure node comprises one or more sub-diagrams (a virtual instrument can be used as a subunit, or sub-diagram, for other virtual instruments, or graphical programs), wherein configuring the event structure node to receive and respond to the one or more user interface events comprises configuring the one or

more sub-diagrams to receive and respond to the one or more user interface events (each virtual instrument graphical program, which can contain sub-diagrams or be a sub-diagram to another graphical program, are configured to produce an output when a certain input is received, for example, when a certain user input is received) (column 8, lines 45-57).

Referring to claims 3, 21 and 34, Kodosky et al. teach for each of the one or more sub-diagrams, configuring the sub-diagram comprises in response to user input, specifying one or more user interface events to which the sub-diagram corresponds, including graphical source code in the sub-diagram in response to user input (such as a plurality of interconnected nodes in the block diagrams) (Figures 16 and 22), wherein the graphical source code is operable to respond to the one or more user interface events to which the sub-diagram corresponds (each virtual instrument graphical program, which can contain a plurality of interconnected nodes, are configured to produce an output when a certain input is received, for example, when a certain user input is received) (column 8, lines 45-57 and column 26, lines 54-64).

Referring to claims 4 and 35, Kodosky et al. teach including two or more interconnected nodes in the sub-diagram (components are connected together via wires) (column 26, lines 54-64 and further shown in Figures 16 and 22).

Referring to claims 5, 51 and 64, Kodosky et al. teach the block diagram comprises a data flow block diagram (column 5, lines 8-9, column 9, lines 65-68 and column 10, lines 1-3). This is further shown in Figure 18.

Referring to claims 6 and 25, Kodosky et al. teach executing the graphical program, wherein one or more user interface events to which the event structure node is configured to receive and respond are generated during execution of the graphical program and the event

structure node is operable to receive and respond to the one or more user interface events generated during execution of the graphical program (when the user clicks on the "GO" button, the graphical program instrument is executed and the outputs are generated) (column 14, lines 10-33 and column 18, lines 29-32).

Referring to claim 7, Kodosky et al. teach the one or more user interface events generated during execution of the graphical program are generated in response to user input to the graphical user interface of the graphical program (when the user clicks on the "GO" button, the graphical program instrument is executed and the outputs are generated) (column 14, lines 10-33 and column 18, lines 29-32).

Referring to claim 10, Kodosky et al. teach configuring the event structure node to receive notification when the one or more user interface events are generated during execution of the graphical program (the virtual instrument block diagram display is notified when one or more of the events have been completed) (column 39, lines 42-47).

Referring to claim 11, Kodosky et al. teach configuring the event structure node to receive information specifying occurrences of the one or more user interface events during execution of the graphical program (the virtual instrument block diagram display are notified, or receive information when one or more of the events have been completed) (column 39, lines 42-47).

Referring to claims 12 and 27, Kodosky et al. teach displaying graphical source code in the event structure node operable to receive and respond to the one or more user interface events (displaying a plurality of interconnected nodes and components in response to user selection and

wiring) (column 8, lines 45-57 and column 26, lines 54-64). This is further shown in Figures 16 and 22.

Referring to claims 13, 46, 48, 62 and 63, Kodosky et al. teach configuring the event structure node to receive and respond to a first graphical user interface event (for example, selection of one of the menu items or buttons shown in Figures 22 and 25 for example), wherein the first graphical user interface event specifies a first user interface element of the graphical user interface and an action performed on the first user interface element (for example, selection of a first user interface element such as the "Clear" menu function causes the block diagram to remove certain components; likewise, selection of the "Reset" button resets the values of the block diagram components) (column 19, lines 11-50).

Referring to claims 14 and 47, Kodosky et al. teach the first user interface element comprises one of an indicator, a control, a menu element, a window (user interface events specify user interface elements such as menu elements used to edit the block diagram and a control such as the "Go" button used to execute the diagram, etc.) (column 18, lines 29-32 and column 19, lines 11-50).

Referring to claims 15, 28, 44 and 60, Kodosky et al. teach receiving user input via a graphical user interface dialog to specify the one or more user interface events (user input selecting the "Special Functions" interface element brings up a dialog box to display the different user interface functions) (column 23, lines 22-26 and further shown in Figure 39).

Referring to claim 16, Kodosky et al. teach displaying an event registration node in the block diagram in response to user input, configuring the event registration node to dynamically register a first user interface event during execution of the graphical program, wherein, after

dynamically registering the first user interface event, the event structure node is operable to receive and respond to the first user interface event (icons and nodes can be registered with particular functions using the dialog box) (column 19, lines 27-61). Furthermore, the user constructs the appropriate execution instructions by constructing an appropriate visual display of interconnected nodes with registered values, which upon execution, will produce a certain output when a corresponding input value is received. In other words, each node, icon and block diagram have execution states that can be triggered by the occurrence of a user interface event; for example, lower level virtual instruments nodes, or sub-diagrams, remain in the "reserved" state until it is dynamically determined during executing, or start of the block diagram virtual instrument display that all higher level nodes have reached an "idle" state (column 14, lines 10-27 and column 36, lines 43-65).

Referring to claim 17, Kodosky et al. teach connecting the event registration node to the event structure node in response to user input (for example, the user can select a node from the Special Functions menu to add and connect to the nodes on the block diagram) (column 23, lines 22-36).

Referring to claim 18, Kodosky et al. teach displaying an event un-registration node in the block diagram in response to user input, configuring the event un-registration node to dynamically un-register a first user interface event during execution of the graphical program, wherein after dynamically un-registering the first user interface event, the event structure node does not receive and respond to the first user interface event (registered values and relationships for nodes can be un-registered, or removed, by using the "CLEAR" function, which removes

selected wires and structures from the block diagram, which un-registers the relationships and values) (column 19, lines 11-15).

Referring to claims 22, 45 and 61, Kodosky et al. teach the one or more programmatic events comprise one or more of a user interface event, a system event, a timer event, an event generated in response to data acquired from a device (events includes user interface events, such as user creating a block diagram by selecting icons to include on the diagram) (column 9, lines 56-64).

Referring to claim 24, Kodosky et al. teach arranging a plurality of nodes on a display and interconnecting the plurality of nodes in response to user input (selecting a plurality of icons, or nodes to include in the block diagram display and connecting the components together via wires) (column 9, lines 56-64, column 26, lines 54-64 and further shown in Figures 16 and 22).

Referring to claim 26, Kodosky et al. teach configuring the block diagram to receive and respond to the one or more user interface events (the user can interact with the block diagram by utilizing icons to build the block diagram; furthermore, as an example, the user interface event "CLEAR", shown in Figure 25, can be used to remove certain wires from the block diagram) (column 9, lines 56-64 and column 19, lines 11-15).

Referring to claim 29, Kodosky et al. teach including an event structure node in the block diagram in response to user input (user designing and interacting with the block diagram, which is made up of connected nodes, or icons, as shown in Figure 22), wherein the event structure node is operable to receive and respond to the one or more user interface events (the user can interact with the block diagram by utilizing icons to build the block diagram; furthermore, as an example, the user interface event "CLEAR", shown in Figure 25, can be used

to remove certain wires from the block diagram) (column 9, lines 56-64 and column 19, lines 11-15). This is further recited in column 8, lines 37-57 and column 50, lines 4-40.

Referring to claim 30, Kodosky et al. teach one or more sub-diagrams (a virtual instrument can be used as a subunit, or sub-diagram, for other virtual instruments, or graphical programs) (column 8, lines 45-57), wherein each sub-diagram includes graphical source code (for example, a plurality of interconnected nodes) (Figures 16 and 22) specifying a response to one or more user interface events (each virtual instrument graphical program, which can contain a plurality of interconnected nodes, are configured to produce an output when a certain input is received, for example, when a certain user input is received) (column 8, lines 45-57 and column 26, lines 54-64).

Referring to claim 31, Kodosky et al. teach a method comprising creating a graphical program (creating a virtual instrument dataflow block diagram) (column 17, lines 48-54), wherein creating the graphical program comprises configuring the graphical program to receive and respond to one or more user interface events (the block diagram contains a plurality of nodes and icons that can receive input and produce the corresponding output) (column 8, lines 45-57), executing the graphical program (selecting the "Go" button executes the virtual instrument block diagram) (column 18, lines 29-32), receiving user input causing generation of a first user interface event, wherein the graphical program is configured to receive and respond to the first user interface event, and sending the first user interface event to the graphical program (when the user selects a button such as "Go" or "Pause", the virtual instrument block diagram receives these interface events and performs the corresponding function; for example, if the user selects the "Clear" function for a particular node or wire, the virtual instrument block diagram display

receives this event and performs the function of removing the selected node or wire) (column 19, lines 11-15). This is further recited in column 50, lines 4-40.

Referring to claims 37, 49 and 54, Kodosky et al. teach the graphical source code executing in response to the user interface event during execution of the graphical program, wherein the user interface event is generated during execution of the graphical program (when the user clicks on the "Go" button, the program is executed, i.e. the block diagrams containing the plurality of interconnected nodes are ran with the inputs causing the corresponding outputs) (column 14, lines 10-27 and column 18, lines 27-32).

Referring to claims 38 and 55, Kodosky et al. teach the graphical source code comprises a plurality of interconnected nodes, as shown by the plurality of connected component in Figures 16 and 22.

Referring to claims 39 and 56, Kodosky et al. teach displaying the plurality of nodes in response to user input and interconnecting the plurality of nodes in response to user input (user input selects the components to display and connect on the block diagram) (column 9, lines 56-64 and column 26, lines 54-64.

Referring to claims 40 and 57, Kodosky et al. teach displaying the graphical source code within the first node in response to user input (displaying subunits containing a plurality of interconnected nodes and icons for components of the block diagram; in other words, each node in the block diagram can contain graphical source code comprising a plurality of interconnected nodes) (column 8, lines 45-57).

Referring to claims 41 and 58, Kodosky et al. teach receiving user input specifying one or more user interface events to which the first node corresponds, wherein the graphical source

code is operable to respond to the one or more user interface events which the first node corresponds (each virtual instrument graphical program, which can contain a plurality of interconnected nodes, are configured to produce an output when a corresponding event is received, for example, when a certain user input is received) (column 8, lines 45-57 and column 26, lines 54-64).

Referring to claims 42 and 59, Kodosky et al. teach associating two or more portions of graphical source code with the first node in response to user input (the user building the block diagram can connect a plurality of subunits containing interconnected components to each of the nodes on the block diagram), wherein each of the portions of graphical source code is operable to respond to one or more of the one or more user interface events (each plurality of interconnected nodes are configured to produce an output when a corresponding event is received, for example, when a certain user input is received) (column 8, lines 45-57 and column 26, lines 54-64).

Referring to claim 43, Kodosky et al. teach receiving user input specifying a name of each of the user interface events (for example, the user interface event with the name of "Clear" can be associated with a particular node to erase, or delete portions or all of the node) (column 15, lines 11-15).

Referring to claim 49, Kodosky et al. teach executing the graphical program, generating the event during execution of the graphical program, wherein executing the graphical program includes executing the graphical source code in response to generating the event.

Referring to claim 50, Kodosky et al. teach the graphical program has a graphical user interface (Figure 22), wherein the method comprises executing the graphical program, generating the user interface event during execution of the graphical program, wherein generating the user

interface event comprises generating the user interface event in response to user input to the graphical user interface, wherein executing the graphical program includes executing the graphical source code in response to generating the user interface event (when the user clicks on the user interface "Go" button, the program is executed, i.e. the block diagrams containing the plurality of interconnected nodes are ran with the inputs causing the corresponding outputs) (column 14, lines 10-27 and column 18, lines 27-32).

Referring to claims 52 and 65, Kodosky et al. teach a plurality of interconnected nodes that visually indicate functionality of the graphical program (column 14, lines 10-27).

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 8-9 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kodosky et al. U.S. Patent 5,301,336 and Zizzo U.S. Patent 6,578,174.

Referring to claims 8 and 9, Kodosky et al. teach all of the limitations as applied to the claims above. Specifically, Kodosky et al. teach the design and display of block diagrams (Kodosky et al.: Figure 22). However, Kodosky et al. fail to explicitly teach the block diagram executes on a first reconfigurable instrument and the graphical user interface is displayed on a display of a second, connected computer system. Zizzo teaches a method for providing a

graphical user interface for creating a graphical program (providing tools for designing circuits) (Zizzo: column 18, lines 36-38) similar to that of Kodosky et al. In addition, Zizzo further teaches the design executes on a first reconfigurable instrument (server computer system) and the graphical user interface is displayed on a display of a second computer system (the circuit design executes on a central server, or first computer system while a plurality of user, or second computer systems, connected to the server through a network displays the user interface used in designing the circuit diagram) (Zizzo: column 4, lines 50-60). It would have been obvious to one of ordinary skill in the art, having the teachings of Kodosky et al. and Zizzo before him at the time the invention was made, to modify system for executing and displaying block diagrams of Kodosky et al. to include the use of a client/server network system in executing and designing diagrams, taught by Zizzo. One would have been motivated to make such a combination in order to allow design tools to be readily available and easily used on a variety of computing platforms and operating systems.

Response to Arguments

4. Applicant's arguments filed on 2 September 2004 have been fully considered but they are not persuasive.

5. Applicant submits that Kodosky nowhere teaches or suggests "...wherein the graphical source code is operable to execute in response to a user interface event...". The examiner respectfully disagrees. Kodosky teaches the display of the block diagram graphical program for a design, which is executed in response to an event such as users configuring the input controls

and clicking on the Go button on the top of the displayed interface, as recited in column 17, line 48 – column 18, line 32; user selection of the Go button, i.e. with the keyboard and mouse connected to the display (column 8, line 58 – column 9, line 30 and Figure 4) is a user interface event and therefore, as shown in Figure 22, the displayed graphical source code program, or graphical block diagram design can be executed when user interface events, i.e. selection of "Go" is received from the user.

6. Applicant also submits that nowhere does Kodosky teach or suggest an "event structure node" where the event structure node is configurable to "receive and respond to one or more user interface events during execution of the graphical program". The examiner respectfully disagrees. The applicant submits that Kodosky's Figure 22 is simply an illustration of an example graphical program that contains no event structure node. Figure 22 is a block diagram for the design example of Figure 21 and shows numerous event structure nodes represented by the virtual instrument icons such as "TEK5010FG", "FLUKE 8840 A", etc., as recited in column 17, line 48 – column 18, line 32. As another example, the block diagrams in Figures 20E – 20L show structure nodes in the diagrams, which upon user configuration of the input controls and selection of the Go button, is operable to receive and respond, or execute during operation of the instrument, as recited in column 17, line 48 – column 18, line 32.

7. Applicant also submits that Kodosky nowhere teaches "creating a graphical user interface for the graphical program in response to user input". The examiner respectfully disagrees. As recited in column 17, line 48 – column 18, line 32, Figure 22 shows a computer-generated

display of a block diagram. Specifically, "the block diagram is the graphical program representing the instrument's operation" (column 17, lines 50-51). The block diagram shown in Figure 22 is displayed on a front panel graphical user interface, as recited in the Abstract and pointed out by the applicant on page 20 of the response to office action filed on 2 September 2004, where Kodosky teaches "a method for programming a computer to execute a procedure, is based on a graphical user interface which utilizes data flow diagrams to represent the procedure". Therefore, the graphical user interface is created, or used to display a graphical program, or block diagram in response to user input such as designing, creating and executing the components of the diagram.

8. Furthermore, the applicant submits that Kodosky does not teach or suggest "...configuring the event structure node to receive and respond to one or more user interface events during execution of the graphical program". The examiner respectfully disagrees. As previously stated, the block diagram displayed in Figure 22 is a graphical program represented by a plurality of event structure nodes. When the block diagram (graphical program) is executed by users configuring the input controls and selecting the Go button, the event structure nodes, or components of the block diagram (graphical program) receives and responds to these user input events by executing, as recited in column 18, lines 27-32, column 34, lines 26-65 and column 38, lines 6-17. The front panel provides interactive use of the virtual instruments, or nodes of the block diagrams, thereby allowing the virtual instruments, or nodes, to receive a specified input variable value and respond by providing a corresponding output variable value, as recited in column 8, lines 37-57.

9. Lastly, the applicant submits that it is nonobvious to combine Kodosky and Zizzo. The examiner respectfully disagrees. Kodosky teaches a graphical user interface for designing and executing block diagrams for virtual instruments, as described in the Abstract. Zizzo similarly teaches a graphical user interface platform for facilitating the design of instruments suchs as circuits and chips, as described in the Abstract and shown in Figures 8a – 8d. Therefore, it would have been obvious to one of ordinary skill in the art to combine the teachings of Kodosky and Zizzo in order to allow improved and easy to use design tools to be readily available and used on a variety of computer platforms; furthermore, this combination facilitates circuit and chip design using shared resources over a distributed electronic network.

10. Therefore, it can be seen that both Kodosky and the combination of Kodosky and Zizzo anticipate the various claimed limitations of the subject invention.

11. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event,

however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ting Zhou whose telephone number is (571) 272-4058. The examiner can normally be reached on Monday - Friday 8:30 am - 6:00 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John Cabeca can be reached at (571) 272-4048. The fax phone number for the organization where this application or proceeding is assigned is (571) 273-4058.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

29 November 2004

JOHN CABECA
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 21